Compressed Dynamic Mesh Sequence for Progressive Streaming

Bailin Yang

Zhaoyi Jiang

Jiantao Shangguan

Frederick W. B. Li

Chao Song

Yibo Guo

Mingliang Xu

Abstract

Dynamic mesh sequence (DMS) is a simple and accurate representation for precisely recording a 3D animation sequence. Despite its simplicity, this representation is typically large in data size, making storage and transmission expensive. This paper presents a novel framework that allows effective DMS compression and progressive streaming by eliminating spatial and temporal redundancy. To explore temporal re-

dundancy, we propose a temporal frame-clustering algorithm to organize DMS frames by their motion trajectory changes, eliminating intra-cluster redundancy by PCA dimensionality reduction. To eliminate spatial redundancy, we propose an algorithm to transform the coordinates of mesh vertex trajectory into a decorrelated trajectory space, generating a new spatially non-redundant trajectory representation. We finally apply a spectral graph wavelet transform (SGWT) with CSPECK encoding to turn the resultant DMS into a multi-resolution representation to support progressive streaming. Experiment results show that our method outperforms several existing methods in terms of storage requirement and reconstruction quality.

Keywords: 3D mesh coding, PCA, 3D mesh sequence compression, progressive streaming, spectrum wavelet transform.

Introduction

Nowadays, 3D animation is widely applied in different domains, including 3D video gaming, film production, 3D imaging and Tele-Medicine. The importance of 3D animation significantly grows due to the rapid development in 3D TV and 3D films. Based on model representation, 3D animation can be classified into 1) animated / dynamic mesh sequence and 2) skeletal animation. Particularly, dynamic mesh sequence (DMS) is crucial to medical applications, as they can precisely capture muscular or pathological changes of organs, which cannot be easily reproduced through skeletal animation. Despite the advantage, data representation of DMS is generally very large in data size, making storage and transmission costly. Hence, research in optimizing DMS representations becomes very popular recently.

Majority of existing work in optimizing DMS mainly focus on performing single resolution compression. This approach is a natural extension of compression methods for static 3D meshes, which assumes the input is a set of vertex information and eliminates data redundancy accordingly. Despite the effectiveness in data reduction, no part of an animation sequence can be reconstructed unless the entire data representation is obtained. This limitation induces significant data transmission and storage requirement bottlenecks to applications, particularly for those requiring access to DMS data stored remotely. To facilitate efficient DMS transmission and rendering, a viable solution is to support progressive transmission of DMS, such that a DMS can be reconstructed even if part of the DMS data is available. Although some work have been done along this line, they generally require re-

meshing, introducing distortions to a DMS and being undesirable to applications that require precise DMS visualization and storage.

This paper proposes a novel method to support effective DMS compression and progressive transmission. We exploit both temporal and spatial redundancy to effectively reduce DMS data size, proposing *temporal frame-clustering* and *spatial decorrelation* algorithms. We also employ spectral graph wavelet transformation (SGWT) [1] with color set partitioning embedded block (CSPECK) encoding [2] to turn resultant DMS into a multi-resolution representation, supporting progressive streaming. Our main contributions include:

- **Temporal frame-clustering**: We extract and organize mesh vertex trajectories of a dynamic mesh sequence into frame-clusters, such that temporal redundancy of mesh motions can be identified and removed by PCA dimensionality reduction.

- **Spatial decorrelation**: We transform coordinates of mesh vertex trajectories into a decorrelated space, identifying independent parts of spatial information about mesh vertex trajectories together with their importance. This allows us to transform mesh vertex trajectories into a compact representation.

- **Progressive coding**: We construct a multi-resolution representation of DMS without re-meshing through spectral graph wavelet transformation with CSPECK encoding. This is particularly important for remote visualization of dynamic mesh sequences under limited bandwidth or multi-user environments.

The rest of the paper is organized as follows. In Section 2, we review relevant existing

work. We present an overview of our proposed framework in Section 3 and elaborate the technical details in Section 4. We compare the performance of our proposed method and existing work in Section 5. Finally, we conclude our work in Section 6.

Related Work

Typically, a mesh is defined by its connectivity and vertex geometry, which describe relations among mesh vertices and specify vertex positions, respectively. Dynamic mesh sequence (DMS) comprises an ordered list of meshes, where each mesh captures an sample of object motion at a specific time, namely a *mesh frame*. Although each mesh in a DMS may possibly be defined with a different connectivity, real-life applications usually impose a fixed connectivity throughout an entire DMS to avoid resolving complicated correspondence among the animated meshes in the DMS. Therefore, our work assumes a DMS is always defined by a fixed connectivity.

Traditional methods: A simple approach to compress a DMS is extending traditional static mesh compression methods to process each mesh frame independently. This essentially only accounts for vertex spatial correlation within each mesh frame, without exploiting any temporal correlation among mesh frames. Compression effectiveness is therefore not impressive, because a major source of DMS data redundancy comes from neighboring mesh frame similarity.

An intuitive approach to eliminate temporal data redundancy of a DMS is adapting ex-

isting video encoding techniques. For example, [3] considered a mesh sequence as a video sequence, transforming each mesh frame into a geometry image (GI) [4], such that the entire DMS can be compressed by conventional video encoding methods. However, this method cannot properly deal with coarse mesh animation models with folding features. This method may also introduce artifacts to a reconstructed DMS due to unavoidable parameterized errors induced in the GI transformation process.

Prediction methods: A most common way for data reduction is to encode only a subset of data, using the encoded data to predict missing information and storing only the prediction error [5–7]. However, this approach was only used to reduce spatial redundancy within a mesh frame. To support both temporal and spatial redundancy reduction, [8] used spatial and temporal predictors [9] to turn input mesh information into compact prediction residuals, and quantized them by arithmetic coding with a context model. This method is too complex for practical implementation. Alternatively, [10,11] proposed a concept of virtual character animation image. This approach applies a fuzzy clustering algorithm to jointly identify data similarity within the anatomy structure of a virtual character model and the temporal coherence within the motion data, in order to facilitate effective data compression.

Decomposition methods: Principal component analysis (PCA) is an operation to transform a dateset into linearly uncorrelated components. It has been used for DMS compression [12,13]. An early work by Alexa and Muller [14] applied singular value decomposition (SVD) to obtain uncorrelated components from a DMS, and that the DMS can be reconstructed using a subset of those components for data reduction. However, when performing

SVD on a very large DMS (in terms of vertex count and number of mesh frames), problems such as memory leaks and missing important components might be encountered. As an extension, [15] exploited temporal coherence of PCA coefficients by encoding them with a linear prediction coding, lowering the entropy of the encoded data. [16] organized data from a DMS into vertex trajectories and clustered them into parts for performing PCA locally, such that fewer PCA components were generated for each part to enhance compression ratio. Alternatively, [17] proposed a lossy method to compress a motion sequence. It applied wavelet transform to encode individual degree-of-freedom (DOF) of a motion sequence, and selected an optimal number of wavelet coefficient to reconstruct each DOF by taking into account human perception. This method worked fine with motion capture or skeletal animation data since they usually possessed small DOFs. The method might run into scalability problem in terms of computation performance when processing DMS in general as DOF of such domain is relatively very large. In addition, the compression ratio achieved by this method was inferior to the PCA-based compression method in [18]. The method by [18] clustered mesh frames by pose similarity and compressed each cluster with PCA in search of a smaller number of PCA components required. It also applied linear predictive coding within each cluster to further reduce output data size. A main issue with this method is that the entire set of clusters must be made available before a DMS can be reconstructed. The best data reduction was achieved by CODDYAC [19], which performed PCA on the trajectory space of a DMS and found a minimal number of significant trajectories characterizing the mesh motion over the sequence. [20] extended [19] to optimize memory storage

requirement by incorporating an average mesh, such that the trajectory coefficients obtained by PCA could be coded with differential coordinates through geometric Laplacians.

Progressive transmission: Recently, some methods [8,21] have been developed based on multi-layer prediction and efficient coding/decoding implementation, respectively. They improved rate-distortion performance to support efficient DMS transmission. However, these methods are not quite scalable to meet dynamic network requirements. The best way to address the problem is by progressive transmission, which allows a coarse DMS representation to be transmitted and reconstructed and finer DMS representations to be gradually reconstructed given more resources are available for transmitting relevant DMS details.

To support progressive transmission, one approach is to design mesh coding adapting network constraints [22–25]. These methods provided reliable progressive transmission of conventional 3D meshes over lossy networks. Another approach is to apply multi-resolution analysis on a DMS. Typically choices include fourier, wavelet and discrete cosine transforms. For instance, [26,27] applied wavelet to generate a multi-level DMS representation along the temporal domain, without considering spatial coherence in each frame. Compression results were not optimal. To improve compression rate, [28–32] exploited spatial and temporal redundancies to compress a DMS with wavelet transform. All these methods were required re-meshing to transform a mesh into a semi-regular form, which was a lossy process and induced mesh distortion. Our work avoids such a distortion problem by adapting spectral graph wavelet transform [33], which was designed to process a mesh of any arbitrary topology, such as full-regular, semi-regular or non-regular.

Overview of Proposed Framework

For a DMS, each frame of the sequence is represented by a 3D mesh describing an object under a particular pose or motion. Meshes of neighboring frames may differ marginally, leading to a large redundancy, namely temporal redundancy. To exploit this characteristic, we may gather such similar frames into an individual cluster to eliminate redundancy. Traditional clustering methods, such as k-means [34], may be employed for this purpose. However, they suffered from local optimality and initialization dependence problems. Also, by performing straightforward clustering to a DMS based on mesh frame similarity, e.g. [18], may not facilitate progressive transmission since the indices of mesh frames assigned to each cluster are not continuous. Alternatively, recent methods [35,36] for generating image superpixels of irregular shapes may also be considered to cluster a DMS. They clustered an image using some image-based constraints, including optimizing commute time and texture measurement or using color and geometric restrictions. However, they imposed area constraints or DBSCAN search range, which limited cluster (superpixel) sizes to be small and compact, being not favorable to our frame-clustering, which is expected to produce arbitrary-sized or noticeable-sized clusters depending on mesh geometry and postures.

To address the problem, we have developed a comprehensive approach to obtain a compact, compressed DMS representation, supporting progressive streaming. We start with a new frame-clustering method based on the curvature of vertex trajectory, efficiently obtaining continuous frames in each cluster. To facilitate compact representation, we apply a

PCA-based method to decorrelate the coordinates information of clustered mesh sequences, identifying and removing spatial redundancy. Vertex trajectories can then be represented using a set of compressed coordinates. We also perform PCA dimensionality reduction to remove intra-cluster temporal redundancy. We further perform spectral graph wavelet transform to eliminate intra-frame redundancy among mesh vertices within a mesh frame. After this, the wavelet coefficients obtained are encoded into a bit-stream using CSPECK, producing a multi-resolution mesh sequence. Note that we adapted CSPECK rather than SPECK because it can handle multi-channel information, i.e. (x,y,z) coordinates, in one go.

Our proposed framework is depicted in Fig. 1 and Fig. 2, showing the workflows of DMS compression and decompression. The main components are:

- **1: Data extraction.** We extract both the geometrical information ($vertex$) and topology information ($face$) from a DMS.

- **2: Frame-clustering.** Temporal redundancy is mainly caused by data similarity along a motion sequence. Exploiting such redundancy can significantly reduce the data size of a DMS, particularly for long motion sequences. To start, we perform clustering to divide a DMS into groups of similar mesh frames. This step avoids us from processing very large data sets, which leads to memory leak problems. It also enables us to improve compression ratio, since such intra-cluster data variance is usually much lower than the entire DMS. Specifically, we divide a DMS into clusters, in which each of them contains a set of continuous frames segmented according to the motion

characteristics of the mesh model. After obtaining the center points of all the mesh frames, we construct a trajectory curve comprising all these center points. We then compute the curvature values at these points along the trajectory curve. We sort the mesh frames by their curvature values, and select the mesh frames corresponding to the top M values as critical frames for classification. With this approach, the indexes of frames including in each cluster will be continuous.

- **3: Mesh signal optimization.** By leveraging the frame-clustering algorithm, we propose a trajectory-based PCA dimensionality reduction algorithm to optimize a DMS based on mesh signals, which are corresponding to individual mesh vertex coordinates representation and the vertex trajectory. Specifically, (x, y, z) coordinates of clustered mesh frame sequences are highly correlated. By PCA analysis, we can determine the main direction of such coordinates under the PCA space, and map the coordinates onto that direction to obtain a compact representation. After constructing vertex trajectory matrices based on these newly generated coordinates, we apply PCA dimensionality reduction to remove temporal redundancy in each cluster.

- **4: SGWT compression.** To eliminate spatial redundancy between neighboring vertices in the same mesh frame, we apply spectral graph wavelet transformation to decompose mesh vertices into orthogonal components. Such transformation is applicable to any DMS since it can process a mesh with arbitrary connectivity. Hence, our method is free from a re-meshing requirement, which is usually a crucial pre-

processing step to existing methods but inducing distortions.

- **5: CSPECK encoding.** Finally, we code the obtained wavelet coefficients by CSPECK, which can support data transmission on different bit-planes from the encoder, facilitating progressive streaming of a DMS.

The decompression process is the inverse of the compression process. We can obtain topological information and spectral wavelet coefficients after decoding the received data. Using the decoded information, PCA coefficients can be obtained by the inverse spectral graph wavelet transformation, thereby, obtaining the corresponding vertex trajectory matrices. Eventually, the complete dynamic mesh sequence can be reconstructed.

Technical Details

Notation

Throughout the rest of the paper, we use the following symbols:

M – input dynamic mesh sequence

F – number of frames in M

N – number of vertices in each frame of M

v_i^f – the i–th vertex of the f-th frame of M

${}_x v_i^f, {}_y v_i^f, {}_z v_i^f$ – x, y, z coordinates of the i-th vertex of the f-th frame of M

$\overline{v_i^f}$ – reconstructed i-th vertex of the f-th frame of M

Frame-clustering Algorithm

Dynamic mesh sequence (DMS) contains significant data redundancy. Early methods usually assume a DMS comprises limited number of frames, and that major source of redundancy is the similarity between neighboring mesh vertices. Due to the advance in motion capture technologies and various needs in recording precise motion sequences, e.g. medical applications, DMS comprising a long animation sequence grows in popularity. Tackling temporal redundancy hence becomes critical.

Major temporal redundancy in a DMS comes from mesh frame similarity. We therefore propose a frame-clustering algorithm to eliminate such redundancy. Our aim is two-fold. While we improve compression rate, we also want to support progressive streaming. So maintaining frame continuity is critical to the compressed representation of a DMS. Consequently, we consider the marginal differences between DMS neighboring frames, organizing them into clusters of similar neighboring frame sequences. We then apply trajectory-based PCA methods to transform each cluster into principal component trajectories and a small number of coefficients. Our frame-clustering algorithm works as follows.

We first obtain DMS motion trajectory. To proceed, we calculate the center point of all mesh frames, generating a central point matrix. Specifically, from the input mesh sequence $M = (M_1, M_2, \cdots, M_F)$, which consists of F frames, we generate the corresponding coordinates matrix sequence $(V^1, V^2, \cdots, V^F)$, where $V^i, i \in [1, F]$ is a $3 \times N$ matrix,

$$V^i = \begin{bmatrix} {}_xv_1^i & {}_xv_2^i & \cdots & {}_xv_N^i \\ {}_yv_1^i & {}_yv_2^i & \cdots & {}_yv_N^i \\ {}_zv_1^i & {}_zv_2^i & \cdots & {}_zv_N^i \end{bmatrix} \tag{1}$$

Let $v_c^i = ({}_{\mathrm{x}}\mathrm{v}_{\mathrm{c}}^{\mathrm{i}}, {}_{\mathrm{y}}\mathrm{v}_{\mathrm{c}}^{\mathrm{i}}, {}_{\mathrm{z}}\mathrm{v}_{\mathrm{c}}^{\mathrm{i}})^{\mathrm{T}}$ be the center point of the i-th mesh frame, defining as:

$${}_xv_c^i = \frac{\sum_{k=1}^{N} {}_xv_k^i}{N}, {}_yv_c^i = \frac{\sum_{k=1}^{N} {}_yv_k^i}{N}, {}_zv_c^i = \frac{\sum_{k=1}^{N} {}_zv_k^i}{N}. \tag{2}$$

We then generate a $3 \times F$ matrix $C = (v_c^1, v_c^2, \cdots, v_c^F)$.

Secondly, we know that the curvature [37] $k(t)$ of a space parametric curve $\overrightarrow{r(t)}$ can be given from the general parameter equation as:

$$k(t) = \frac{\|\overrightarrow{r'(t)} \times \overrightarrow{r''(t)}\|}{\|\overrightarrow{r'(t)}\|^3}, \overrightarrow{r'(t)} \neq 0 \tag{3}$$

Hence, when $\overrightarrow{r} = (x(t), y(t), z(t))$, the curvature $k(t)$ can be expressed as:

$$k(t) = \frac{\|(x'(t), y'(t), z'(t)) \times (x''(t), y''(t), z''(t))\|}{\|(x'(t), y'(t), z'(t))\|^3} \tag{4}$$

From (4), we can observe that we need to calculate the first and second derivatives of the center points in the x-, y-, and z-directions. We use the finite differences to evaluate the first and second derivatives of the displacement vector. The approximations of the derivatives using central differences [38] are given by:

$$u'(i) = \frac{u(i+1) - u(i-1)}{2\triangle t}$$
$$u''(i) = \frac{u(i+1) - 2u(i) + u(i-1)}{\triangle t * \triangle t} \tag{5}$$

We then calculate the derivative of each 3D discrete point by this method. The central difference method is applied on each vertex in the x-, y-, and z-directions, respectively. When computing the derivative of the center point of i-th frame in each coordinate direction, we use the coordinates of frame $i-1$ and frame $i+1$.

The first derivatives in x-, y-, and z-directions are given by:

$$x(i) = \frac{C(1, i+1) - C(1, i-1)}{2}$$
$$y(i) = \frac{C(2, i+1) - C(2, i-1)}{2} \tag{6}$$
$$z(i) = \frac{C(3, i+1) - C(3, i-1)}{2}$$

The second derivatives in x-, y-, and z-directions are given by:

$$xx(i) = C(1, i+1) - 2 * C(1, i) + C(1, i-1)$$
$$yy(i) = C(2, i+1) - 2 * C(2, i) + C(2, i-1) \tag{7}$$
$$zz(i) = C(3, i+1) - 2 * C(3, i) + C(3, i-1)$$

where $C(i, j)$ indicates the j-th element in the i-th row of the matrix C.

With this approach, we obtain the first and second derivatives of each center point in each of the x-, y-, and z-directions. Combining the derivative information with the above curvature estimation formula, we obtain the curvature of the trajectory curve at each point.

Finally, we sort the frames by their curvature values and select critical mesh frames corresponding to the largest curvature values, and they are regarded as the boundary of

neighbor clusters for classification. Therefore, the indices of the frames included in each cluster will be continuous. Our method can obtain the frame clustering results quickly and accurately. We can also ensure the mesh frames in the same cluster exhibit similar motion trend, improving the efficiency of PCA process in the next step.

Mesh Signal Optimization

By leveraging the results from frame-clustering, we adapt PCA to eliminate redundancy from a DMS by optimizing the mesh signal comprising the DMS. There are two levels of mesh signal, namely the vertex coordinate information and the vertex trajectory. Considering data is highly correlated in three coordinates, we perform a decorrelation operation, obtaining a new independent x-, y-, and z-coordinates representation. Based on the result of the proposed frame-clustering algorithm, we apply PCA dimensionality reduction on each cluster after building trajectory matrices with the new coordinates representation.

Technically, after frame-clustering, the original DMS can be expressed as:

$$\begin{aligned} M = \{M_1, \cdots, M_F\} = \{C_1, \cdots, C_K\} = \{(M_{11}, \cdots, \\ M_{1m_1}), (M_{21}, \cdots, M_{2m_2}), \cdots, (M_{K1}, \cdots, M_{Km_K})\} \end{aligned} \tag{8}$$

where $C_k = \{M_{k1}, M_{k2}, \cdots, M_{km_k}\}, k \in [1, K]$ represents the k-th cluster and the frames contained, m_k denotes the number of frames in cluster C_k. In the following, we consider C_k as an example to describe how PCA dimensionality reduction works.

Step 1: Decorrelate x-, y-, and z-coordinates.

(1) We construct three $m \times N$ vertex trajectory matrices of the k-th cluster in x-, y-, and z-directions as follows:

$$
\begin{aligned}
{}_xT_k &= ({}_xt_1, {}_xt_2, \cdots, {}_xt_N) = \begin{pmatrix} {}_xV^{k1} \\ {}_xV^{k2} \\ \cdots \\ {}_xV^{km} \end{pmatrix} = \begin{pmatrix} {}_xv_1^{k1} & \cdots & {}_xv_N^{k1} \\ \cdots & \cdots & \cdots \\ {}_xv_1^{km} & \cdots & {}_xv_N^{km} \\ & & \end{pmatrix} \\
{}_yT_k &= ({}_yt_1, {}_yt_2, \cdots, {}_yt_N) = \begin{pmatrix} {}_yV^{k1} \\ {}_yV^{k2} \\ \cdots \\ {}_yV^{km} \end{pmatrix} = \begin{pmatrix} {}_yv_1^{k1} & \cdots & {}_yv_N^{k1} \\ \cdots & \cdots & \cdots \\ {}_yv_1^{km} & \cdots & {}_yv_N^{km} \\ & & \end{pmatrix} \cdots \qquad (9) \\
{}_zT_k &= ({}_zt_1, {}_zt_2, \cdots, {}_zt_N) = \begin{pmatrix} {}_zV^{k1} \\ {}_zV^{k2} \\ \cdots \\ {}_zV^{km} \end{pmatrix} = \begin{pmatrix} {}_zv_1^{k1} & \cdots & {}_zv_N^{k1} \\ \cdots & \cdots & \cdots \\ {}_zv_1^{km} & \cdots & {}_zv_N^{km} \\ & & \end{pmatrix}
\end{aligned}
$$

where each column of the above three matrices represents a vertex trajectory coordinates in cluster K.

The resulting center coordinates matrix is given by:

$$
P_k = \begin{bmatrix} {}_xp_1 & {}_xp_2 & \cdots & {}_xp_N \\ {}_yp_1 & {}_yp_2 & \cdots & {}_yp_N \\ {}_zp_1 & {}_zp_2 & \cdots & {}_zp_N \end{bmatrix} \qquad (10)
$$

(2) We obtain a 3×3 matrix U_k, which is regarded as a transformation matrix by applying PCA on center coordinates matrix P_k. We then compute the new vertex coordinates M'_k using the coordinate transformation $(V^{k_j})' = U_k \cdot V^{k_j}$. For the sake of clarity, in the following, V^{k_j} denotes the new vertex matrix of the k_j-th frame.

Step 2: Construct three trajectory matrices in x-, y-, and z-directions and apply PCA to reduce redundancy.

We perform PCA decomposition on the three vertex trajectory matrices and integrate the resulting PCA coefficients. We demonstrate the algorithm by considering ${}_xT_k$ as an example and the subscript label x can be omitted without loss of generality.

We first compute the average value t_a by considering the trajectory of each vertex as a signal sample, obtaining $t_a = (t_1 + t_2 + \cdots + t_N)/N$.

We then compute a new trajectory matrix by subtracting the mean value from the original trajectory matrix as $T'_k = T_k - repmat(t_a, 1, N)$. We consider the set of eigenvectors $E = (e_1, \cdots, e_m)$ obtained from the eigendecomposition of matrix $T_k T'_k$, as a set of bases in the trajectory space. Thus, we can use a new set of optimized coefficients to describe the trajectory space. We order the eigenvectors according to the corresponding eigenvalues. The higher the eigenvalue, the corresponding eigenvector is much dominant. In our method, we only select the p eigenvectors $e_1, \cdots, e_p, p < m$ corresponding to the largest eigenvalues, forming of the new set of bases. Consequently, we can achieve the aim of reducing the dimension of correlated coefficients by using a fewer number of mapping coefficients.

Given the set $B = (e_1, \cdots, e_p)$, the trajectory of vertex v_i in the x-axis can be rep-

resented by a linear combination of the bases in B and the combining coefficients can be composed of PCA coefficients vector $s_i = B^T(t_i - t_a) \in R^p, i = 1, \cdots, N$. Since the data in the same cluster have a high redundancy, the number of basis vectors p could be much smaller than the number of frames in the cluster.

To summarize, we only need to encode the characteristic matrix B, the PCA coefficients vector s_i after dimension reduction and the average trajectory t_a. In this work, we use the arithmetic coding principle to encode the characteristic matrix B and t, as they comprise a small amount of data. On the other hand, we consider the PCA coefficients as signal points defined on the mesh, and perform spectral graph wavelet transform (SGWT). In addition, we use $\overline{B}$, $\overline{s_i}$ and $\overline{t}$ to represent the corresponding reconstruction data. Therefore, trajectory reconstruction of the i-th vertex is represented as $\overline{t_i} = \overline{B} \cdot \overline{s_i} + \overline{t_a}$.

Spectral Wavelet Transform and CSPECK Encoding

Wavelet transform is widely used in data compression. Here, we review some basic definitions about spectral graph theory [33] and show how we extend its application to construct spectral graph wavelets for compressing a DMS.

Consider the generated PCA coefficients as signal points defined on a mesh, we perform SGWT to obtain a set of spectral wavelet coefficients. We compress the 3D mesh model by encoding these coefficients. In order to make our work scalable to a limited network bandwidth environment, we support progressive streaming of DMS by implementing CSPECK

encoding. The spectral wavelet coefficients can then be progressively transmitted with their low frequency components first followed by high frequency components gradually.

After the dimension reduction, all of the trajectory vector s_i in cluster k form an $p \times N$ matrix S. It is regarded as an undirected, weighted graph $G = \{V, E, W\}$ with $|V| = N$ where V, E, W represent the set of vertex signals, edges and weights on edge, respectively. The graph is assumed to be connected. Also, s_i is related to one node of the graph and each row of the matrix S can be regarded as graph signal defined on the vertex of a mesh.

The unnormalized Laplacian is defined as $L = D - A$, where the off-diagonal elements of the degree matrix D is a diagonal matrix. The diagonal elements of D correspond to the degree of each vertex. The degree of each vertex is the sum of the weights of its incident edges. The matrix A is the adjacency matrix of the graph.

We denote the set of orthonormal eigenvectors of L and their associated real eigenvalues by χ_l and λ_l for $L\chi_l = \lambda_l\chi_l, l = 1, 2, \cdots, N$, respectively. Without loss of generality, we assume the eigenvalues of the Laplacian of the connected graph are ordered, i.e., $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. The graph Fourier transform on the vertices of graph G is then defined as:

$$\widehat{f}(l) = \langle \chi_l, f \rangle = \sum_{n=1}^{N} \chi_l^*(n) f(n) \tag{11}$$

and the corresponding inverse transform is given by:

$$f(n) = \sum_{l=1}^{N} \hat{f}(l) \chi_l(n) \tag{12}$$

The spectral graph wavelet transform is generated by wavelet operators that are operator-valued functions of the Laplacian. The transform is determined by the choice of a kernel

function g, which is analogous to the Fourier transform of a wavelet in the classical setting. This kernel g behaves as a band-pass filter, satisfying $g(0) = 0$ and $\lim_{x\to\infty} g(x) = 0$. The wavelet operator $T_g = g(L)$ is defined through its action on a given function f as $\widehat{T_g f(l)} = g(\lambda_l \widehat{f}(l))$. From the spectral theory, we have $g(sL)\chi_l = g(s\lambda_l)\chi_l$. So the wavelet operator at scale s is then defined as $T_g^s = g(sL)$ and the spectral graph wavelet function of scale s placed at vertex n is defined by $\psi_{s,n} = T_g^s \delta_n$. The wavelet coefficients of a function f are computed by taking the inner products with the wavelets:

$$W_f(s,n) =< \psi_{s,n}, f >= (T_g^s f)(n) = \sum_{l=1}^{N} g(s\lambda_l)\widehat{f(l)}\chi_l(n) \tag{13}$$

The SGWT also includes a second class of waveforms called scaling functions, which are analogous to the low-pass residual scaling functions from classical wavelet analysis. In order to stably represent the low frequency content of signals defined on the vertices, they are constructed in a manner analogous to the wavelets, with the scaling function at vertex n defined by $\phi_n = T_h \delta_n$ and the coefficients by:

$$S_f(n) = \langle \phi_n, f \rangle = (T_h f)(n) = \sum_{l=1}^{N} h(\lambda_l)\widehat{f(l)}\chi_l(n) \tag{14}$$

The scale kernel h acts as a low-pass filter, satisfying $h(0) = 0$ and $\lim_{x\to\infty} h(x) = 0$.

To summarize, given a fixed set of wavelet scales $\{s_j\}_{j=1}^{J}$, the wavelet generator g and the scaling generator h, the overall spectral graph wavelet transform is a linear map W defined by:

$$Wf = ((T_h f)^T, (T_g^{s_1} f)^T, \cdots, (T_g^{s_J} f)^T)^T \tag{15}$$

This operator can be shown to be equivalent of a frame, with frame boundaries A and B, which can be estimated by knowing h, and an upper bound on the spectrum of L. It can then be shown that for any function f, we have the following $A\|f\|^2 \leq \|Wf\|^2 \leq B\|f\|^2$, where $A = minG(\lambda), B = minG(\lambda), \lambda \in [0, \lambda_N]$ and $G(\lambda) = h^2(\lambda) + \sum_{i=1}^{n} g^2(s_i\lambda)$.

In this work, the Chebyshev polynomial approximation method is used to obtain the wavelet kernel function. Setting the number of scales $J = 4$, we obtain a cell matrix including five coefficient matrices by definition.

In the color image coding method, the SPECK operations for the 2D wavelet transform coefficients of the color planes are interleaved in each pass. Similarly, we adopt CSPECK for coding 1D coefficient vectors of arbitrary size by interleaving the SPECK operations for the three 1D coefficient vectors of the three coordinates in each pass. We use a recursive set-partitioning procedure to cluster the energy in frequency and space.

For each coordinate $\gamma \in x, y, z$, the coefficients are ordered according to their scales (in the increasing order) and are arranged from high to low energy. No overhead bits are required to inform the decoder about the configuration. In this work, CSPECK consists of multiple coding passes and operates on coding the long 1D coefficient vectors of space coordinates, which is composed of the three different vectors in an interleaved manner.

Note that, in order to realize multi-resolution reconstruction of a DMS, we encode high and low frequency information separately. When network bandwidth is insufficient, we only transmit low frequency information to the client and reconstruct a low resolution DMS. When the network bandwidth increases, we transmit high frequency information gradually,

such that progressive streaming of a DMS can be achieved.

Results

In this section, we present the experiment results of our framework. In the experiments, following parameters are varied to study their impact to the performance of our algorithms:

- Number of clusters K used in frame-clustering.
- Number p of the most significant vectors used in PCA.
- Number of scales J when performing SGWT.
- Number of bit-planes sm and the number of quantization bits q used in CSPECK.

The optimal values of K and p for different models can only be obtained by performing a large number of experiments. Hence, we do not provide additional discussion regarding the optimal choice of parameter values for K and p. From the results, it is observed that the influence of the parameter sm on the final compression ratio is more pronounced in comparison with the parameter q. In addition, for the same value of sm, varying the number of quantization bits in the range of $6 \sim 12$ did not significantly influence the compression efficiency. Hence, we run experiments by setting different sm values. We also fixed $J = 4$ in all the experiments.

Typically, vertex-based error measures, such as root-mean-square or Karni and Gotsman [39] error are used for evaluating the performance of a compression algorithm. In this work,

we choose KG error as a performance metric, which is defined as follows:

$$KG_{error} = 100 \times \frac{\|A - A'\|}{A - E(A)}\% \tag{16}$$

where A is a $3N \times F$ vertex coordinate matrix of original DMS, with N and F denoting the number of vertices and frames, respectively. Each column represents a frame of animation. Matrix A' is a reconstruction data matrix having the same dimensions as matrix A. Also, matrix E represents the per-frame averages of vertex coordinate matrix, and has the same dimensions as matrix A. The reconstruction error is calculated using the Frobenius norm.

In addition, we evaluate bitrate R in bpfv (bits per frame and vertex), which is obtained from the total number of bits Q required to encode the DMS, as:

$$R = \frac{Q(bit)}{N \cdot F} \tag{17}$$

Data of Mesh Sequence

We select mesh sequences of cow, $horse$, $chicken$ and $dance$ for testing the proposed compression algorithm. Relevant data of these animations are shown in Table 1.

Results of Temporal Frame-clustering

Considering cow and $dance$ as examples, the curvature value distributions and the clustering results are shown in Fig. 3. From Fig. 3(a), it can be observed that dividing the cow sequence into 6 segments is optimal. Therefore, we selected mesh frames corresponding to the top 6

values as critical frames for classification, and the corresponding frame-clustering result is shown in the upper right corner of Fig. 3(a). By further testing with different segments, it is also observed that dividing into 6 segments resulted in an optimal compression ratio.

However, Fig. 3(b) shows that, for the $dance$ sequence, there are several large curvature values between frames 160-180. If we only consider the curvature values when selecting segmentation frames, this will fall into a local optimum. Hence, we need to use global curvature value distribution and sort the values to get the final clustering result. Corresponding result is shown in the upper right corner of Fig. 3(b).

Multi-Resolution Reconstruction of DMS

In order to realize multi-resolution reconstruction of a DMS, we used a spectral wavelet transform which is directly defined on the mesh. Hence, the high and low frequency components can be encoded and transmitted separately to realize both data compression and progressive streaming of a DMS. When the network bandwidth is low, we only transmit low frequency information to the client. When bandwidth increases, we transmit high frequency information gradually. Using the first cluster $C1$ (Frame 1 $\sim$ Frame 5) of the cow animation as an example, we choose $J = 4$ and then the coefficient matrix $c = (C_0^T, C_1^T, C_2^T, C_3^T, C_4^T)^T$ can be obtained after spectral wavelet transform, where C_0^T is a scale coefficient vector, and $C_1^T \sim C_4^T$ are wavelet coefficient vectors. By choosing scale coefficient with different layers of wavelet reconstruction, we can obtain different resolutions of a DMS as illustrated in

Fig. 4. With the layer of wavelet reconstruction increases, the detail of reconstructed mesh will be richer, e.g. the feet and breast parts.

Analysis and Comparison with Relevant Algorithms

Now, we compare the performance of the proposed compression algorithm with various existing algorithms in terms of rate vs distortion curves. For this purpose, we computed the KG error and bit rate R for different values of sm.

First, for the dynamic mesh sequences considered in this work, the results of the KG error, bit rate R, and data compression ratio values are shown in Table 2 and Fig. 5. Results showed that our proposed method produced better compression performance.

Second, our proposed method has been compared with various existing algorithms. For the cow, $dance$, $horse$ and $chicken$ animation sequences, we compared the performance of the proposed algorithm with the CODDYAC algorithm [19], Luo's algorithm [18], which also used the temporal segmentation, the ORLPCA algorithm [40] and the optimized mesh traversal for dynamic mesh compression algorithm [41]. The results of the performance comparison are shown in Fig. 6.

As shown in Fig. 6, for the cow, $horse$ and $chicken$ animation sequences, the proposed compression algorithm has a better performance comparing with other four algorithms. Nevertheless, due to the rotation motion in the $dance$ animation, our proposed method cannot obtain a more accurate clustering result. Consequently, our proposed method cannot get

an ideal PCA dimension reduction and compression effect. As shown in Fig. 6(d), only when the value of bit rate R is lower than 0.5, the KG error of the proposed method will be lower than other four algorithms with the same value of R. Therefore, the proposed compression method is limited by the motions in a DMS.

To conclude, after testing against several classical methods, we demonstrated that our proposed approach can obtain a better compression rate. Comparing with existing classical as well as latest compression methods, our approach surpasses them in terms of requiring a smaller storage space and inducing less reconstruction error.

Conclusions and future work

This work proposes a novel three-dimensional dynamic mesh sequence compression framework, which is suitable for progressive transmission, particularly fitting well with collaborative virtual environments [42]. The method first implements a temporal frame-clustering algorithm based on the curvature of vertex trajectory. It provides better clustering result, such that a PCA dimension reduction can be applied on the vertex trajectory coordinates to generate a new and compact representation. In addition, a new theory of spectral wavelet transform has been used for dynamic mesh data compression. After obtaining the spectral wavelet coefficients, low and high frequency components of dynamic mesh data can be encoded and transmitted separately for realizing progressive streaming.

Although the proposed compression algorithm has many advantages, there are still prob-

lems remained to be solved, such as the adaptive progressive transmission of mesh sequence and the perceptual metrics focus on local relations. For future work, we plan to use the STED metric [19] to measure our results. Also, we would like to consider using deep learning methods to detect saliency [43] of a DMS, formulating a method to improve DMS compression rate.

Acknowleogement

This work was partly supported by National Natural Science Foundation of China (Grant Nos. 61472363,61572436) and ZheJiang Province Natural Foundation (Grant No. LY16F020001).

References

[1] J. D. J. G. Leandro, R. M. C. Junior, and R. S. Feris, "Shape analysis using the spectral graph wavelet transform," in *IEEE International Conference on Escience*, 2013, pp. 307–316.

[2] S. F. Lin, H. C. Hsin, and C. K. Su, "Hybrid image compression based on set-partitioning embedded block coder and residual vector quantization," *Journal of Information Science & Engineering*, vol. 26, no. 3, pp. 1011–1027, 2010.

[3] M. B. Hector, V. S. Pedro, M. Leonard, G. Steven, and H. Hugues., "A new representation for 3d animations," *ACM SIGGRAPH/Eurographics symposium on Computer animation*, vol. 03, pp. 136–146, 2003.

[4] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 355–361, 2002.

[5] J. E. Lengyel, "Compression of time-dependent geometry," *Proceedings of Symposium on Interactive 3D graphics*, pp. 89–95, 1999.

[6] L. Vasa and V. Skala, "Driven local neighbourhood based predictors for dynamic mesh compression," *Computer Graphics Forum*, vol. 29, no. 6, pp. 1921–1933, 2010.

[7] M. Xu, M. Li, W. Xu, Z. Deng, Y. Yang, and K. Zhou, "Interactive mechanism modeling from multi-view images," *ACM Transactions on Graphics*, vol. 35, no. 6, p. 236, 2016.

[8] J.-K. Ahn, Y. J. Koh, and C.-S. Kim, "Efficient fine-granular scalable coding of 3d mesh sequences," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 485–497, 2013.

[9] L. Ibarria and J. Rossignac, "Dynapack:space-time compression of the 3d animations of triangle meshes with fixed connectivity," in *ACM Symp. Computer Animation*, 2003, pp. 126–135.

[10] B.-S. Chew, L.-p. Chau, and K.-H. Yap, "A fuzzy clustering algorithm for virtual character animation representation," *IEEE Transactions on Multimedia*, vol. 13, no. 1, pp. 40–49.

[11] M. Xu, J. Zhu, P. Lv, B. Zhou, M. F. Tappen, and R. Ji, "Learning-based shadow recognition and removal from monochromatic natural images." *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5811–5824, 2017.

[12] S.-Z. Li, B. Yu, W. Wu, S.-Z. Su, and R.-R. Ji, "Feature learning based on sae–pca network for human gesture recognition in rgbd images," *Neurocomputing*, vol. 151, pp. 565–573, 2015.

[13] E. Oja, "The nonlinear pca learning rule in independent component analysis," *Neurocomputing*, vol. 17, no. 1, pp. 25–45, 1997.

[14] M. Alexa and W. Muller, "Representing animations by principal components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[15] L. Ibarria and J. D. Rossignac, "space-time compression of the 3d animations of triangle meshes with fixed connectivity," *Proceedings of ACM SIGGRAPH/ Eurographics symposium on Computer animation*, pp. 126–135, 2003.

[16] M. Sattler, R. Sarlette, and R. Klein, "Simple and efficient compression of animation sequences," in *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2005, pp. 209–217.

[17] A. Firouzmanesh, I. Cheng, and A. Basu, "Perceptually guided fast compression of 3-d motion capture data," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 829–834, 2011.

[18] G. Luo, F. Cordier, and H. Seo, "Compression of 3d mesh sequences by temporal segmentation," *Computer Animation and Virtual Worlds*, vol. 24, no. 34, pp. 365–375, 2013.

[19] L. Vasa, V. Skala, and Coddyac, "Connectivity driven dynamic mesh compression," *3DTV-CON, The True Vision - Capture, Transmission and Display of 3D Video, IEEE Computer Society, Kos,Greece*, 2007.

[20] L. Váša, S. Marras, K. Hormann, and G. Brunnett, "Compressing dynamic meshes with geometric laplacians," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 145–154.

[21] R. Mekuria, M. Sanna, E. Izquierdo, D. C. Bulterman, P. Cesar *et al.*, "Enabling geometry-based 3-d tele-immersion with fast mesh compression and linear rateless coding," *IEEE Transactions on Multimedia*, vol. 16, no. 7, pp. 1809–1820, 2014.

[22] G. AlRegib, Y. Altunbasak, and J. Rossignac, "An unequal error protection method for progressively transmitted 3d models," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 766–776, 2005.

[23] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Optimal packet loss protection of progressively compressed 3-d meshes," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1381–1387, 2009.

[24] M. Xu, C. Li, P. Lv, N. Lin, R. Hou, and B. Zhou, "An efficient method of crowd aggregation computation in public areas," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[25] D. Tian and G. AlRegib, "Multistreaming of 3-d scenes with optimized transmission and rendering scalability," *IEEE Transactions on Multimedia*, vol. 9, no. 4, pp. 736–745, 2007.

[26] F. Payan and M. Antonini, "Wavelet-based compression of 3d mesh sequences," *Proceedings of ACIDCA-ICMI'2005*, 2005.

[27] Y. Boulfani-Cuisinaud and M. Antonini, "Motion-based geometry compensation for dwt compression of 3d mesh sequences," in *IEEE International Conference on Image Processing*, vol. 1. IEEE, 2007, pp. I–217.

[28] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2004, pp. 183–192.

[29] J.-W. Cho, M.-S. Kim, S. Valette, H.-Y. Jung, and R. Prost, "3-d dynamic mesh compression using wavelet-based multiresolution analysis," in *IEEE International Conference on Image Processing*. IEEE, 2006, pp. 529–532.

[30] J. W. Cho, S. Valette, J. H. Park, H. Y. Jung, and R. Prost, "3-d mesh sequence compression using wavelet-based multi-resolution analysis," *Applied Mathematics and Computation*, vol. 216, no. 2, pp. 410–425, 2010.

[31] N. Stefanoski and J. Ostermann, "Spc: fast and efficient scalable predictive coding of animated meshes," in *Computer Graphics Forum*, vol. 29, no. 1. Wiley Online Library, 2010, pp. 101–116.

[32] J.-W. Cho, M.-S. Kim, S. Valette, H.-Y. Jung, and R. Prost, "A 3-d mesh sequence coding using the combination of spatial and temporal wavelet analysis," in *Computer Vision/Computer Graphics Collaboration Techniques*. Springer, 2007, pp. 389–399.

[33] K. H. David, V. Pierre, and G. Remi, "Wavelets on graphs via spectral graph theory original," *Research Article Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, March 2011.

[34] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[35] J. Shen, Y. Du, W. Wang, and X. Li, "Lazy random walks for superpixel segmentation." *IEEE Transactions on Image Processing*, vol. 23, no. 4, pp. 1451–1462, 2014.

[36] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by dbscan clustering algorithm," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5933–5942, 2016.

[37] *Curvature Equation*. Springer Berlin Heidelberg, 2009.

[38] L.-I. W. Roeger and R. W. Barnard, "Preservation of local dynamics when applying central difference methods: application to sir model," *Journal of Difference Equations & Applications*, vol. 13, no. 4, pp. 333–340, 2007.

[39] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25–34, 2004.

[40] A. Rachida and S. Wolfgang, "Efficient compression of 3d dynamic mesh sequences," *Journal of the WSCG*, pp. 99–107, 2007.

[41] L. Vasa, V.Skala, and Coddyac, "Optimized mesh traversal for dynamic mesh compression," *Graphical Models*, vol. 73, no. 2, pp. 218–230, 2011.

[42] F. W. B. Li, R. W. H. Lau, D. Kilis, and L. W. F. Li, "Game-on-demand:: An online game engine based on geometry streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 7, no. 3, Sep 2011.

[43] B. Yang, F. W. B. Li, X. Wang, M. Xu, X. Liang, Z. Jiang, and Y. Jiang, "Visual saliency guided textured model simplification," *The Visual Computer*, vol. 32, no. 11, pp. 1415–1432, Nov 2016.

Table 1: Data of Models

Name	cow	horse	chicken	dance
Num of vertices	2904	1000	3030	7061
Num of faces	5804	1990	5664	14118
Num of frames	204	200	400	201
Geometry information(KB)	12724	5704	19064	30222
Topology information(KB)	21	9	22	48

Table 2: The compression results with different models

Name	sm	KG error(%)	Rate(bpfv)	After(KB)	ratio
cow	2	0.2700	1.2099	88	1:145.66
	4	0.0691	2.6757	194	1:65.97
	6	0.0619	5.7364	415	1:30.72
	8	0.0582	11.1178	804	1:15.85
horse	2	0.3911	0.8225	31	1:181.13
	4	0.0806	1.4334	55	1:103.94
	6	0.0536	2.41848	94	1:61.60
	8	0.0521	4.19145	164	1:35.54
chicken	2	0.4268	0.7484	130	1:143.19
	4	0.0989	1.9750	342	1:55.78
	6	0.0516	4.6032	798	1:23.93
	8	0.0233	8.6128	1492	1:12.79
dance	2	0.2727	0.8687	151	1:201.13
	4	0.0859	1.8586	322	1:94.01
	6	0.0811	3.7470	649	1:46.629
	8	0.0808	5.3564	928	1:32.619

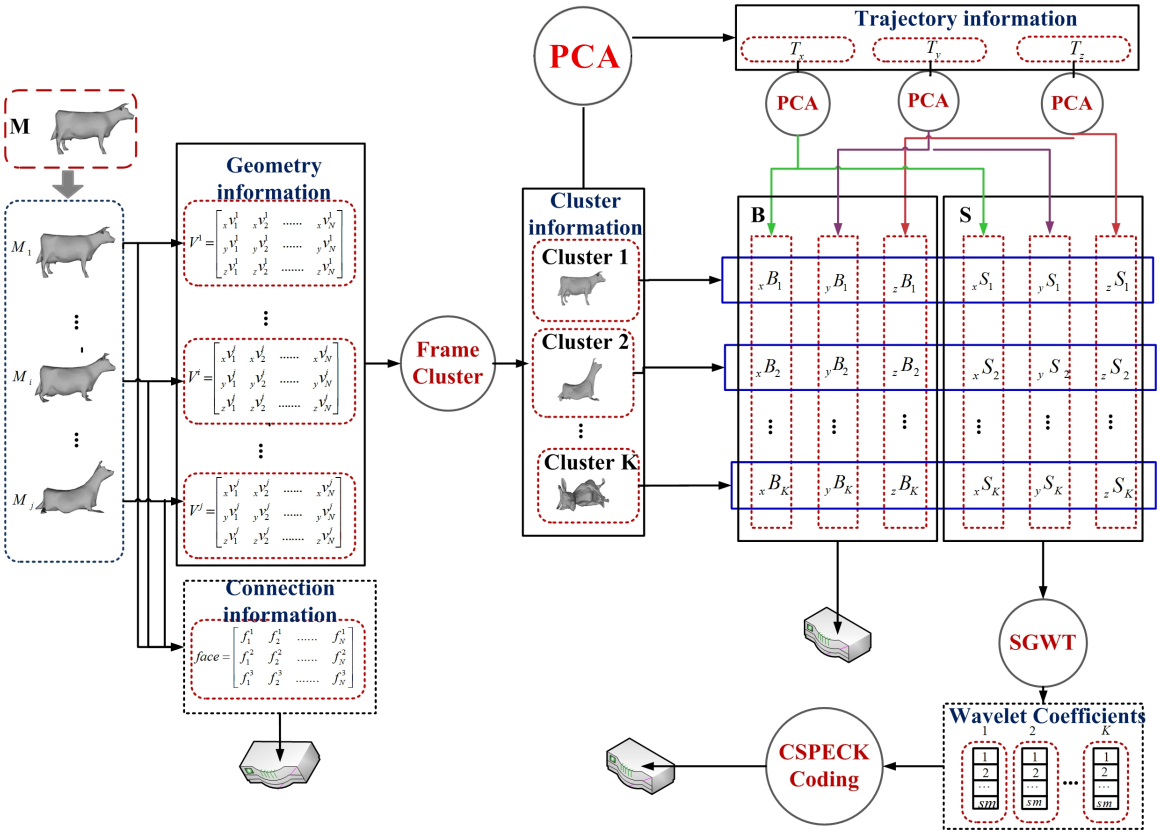


Figure 1: Work flow of the proposed compression algorithm

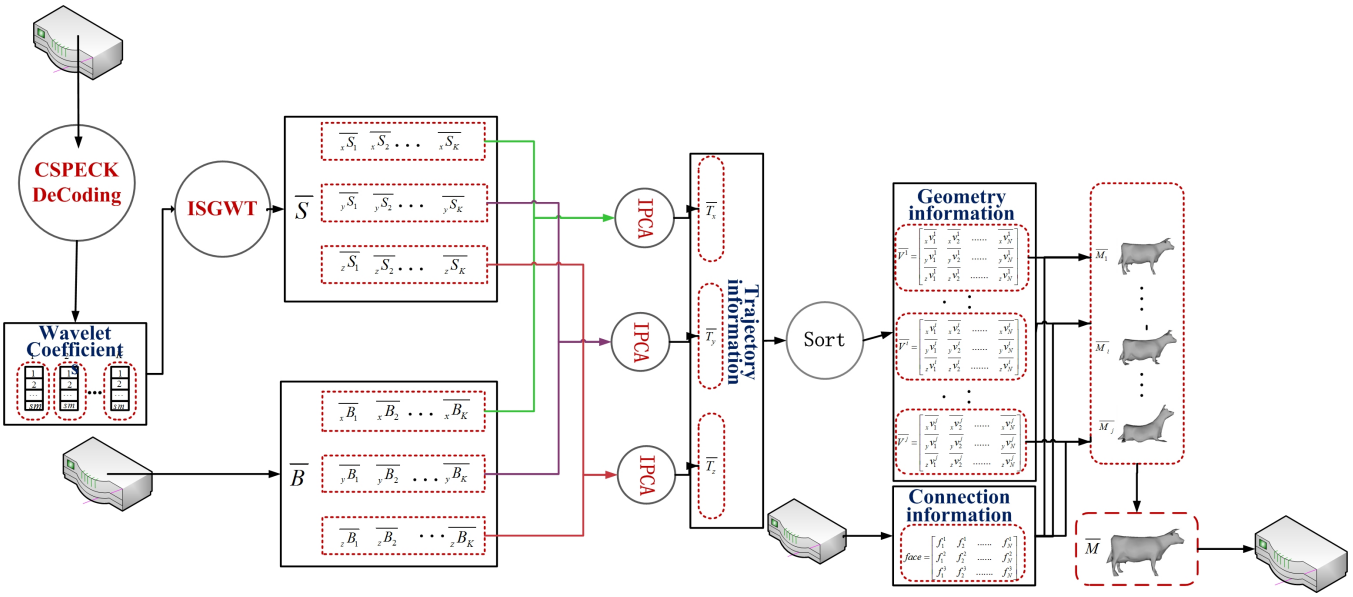
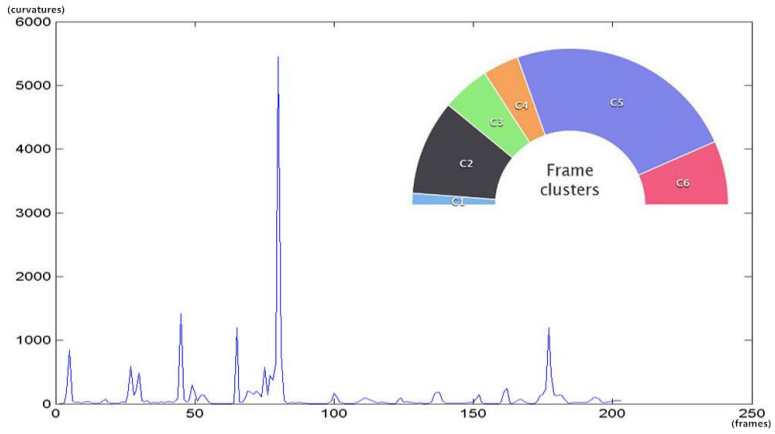
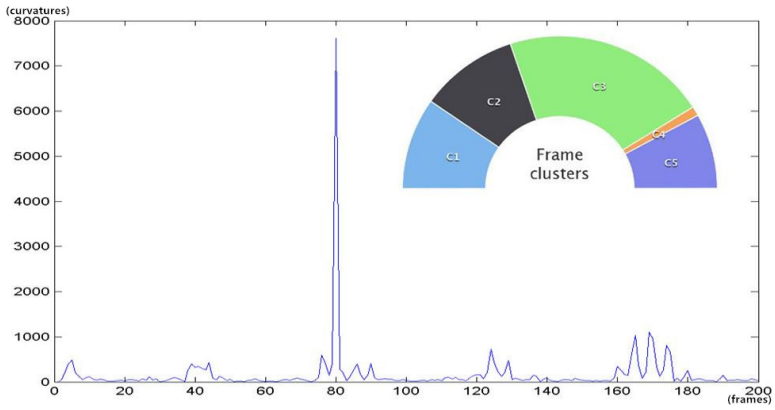


Figure 2: Work flow of the proposed decompression algorithm

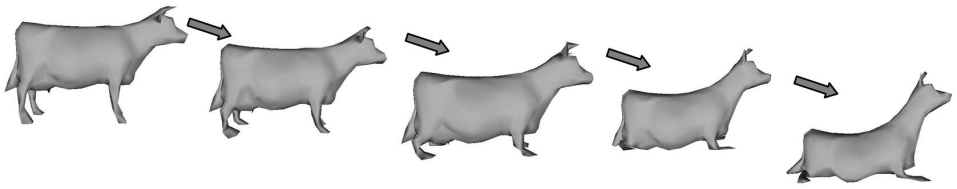


(a) cow: curvature value distribution and clustering result

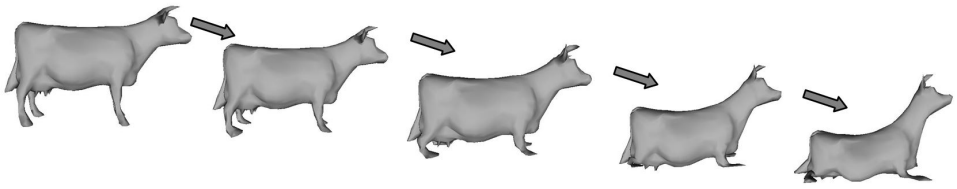


(b) dance: curvature value distribution and clustering result

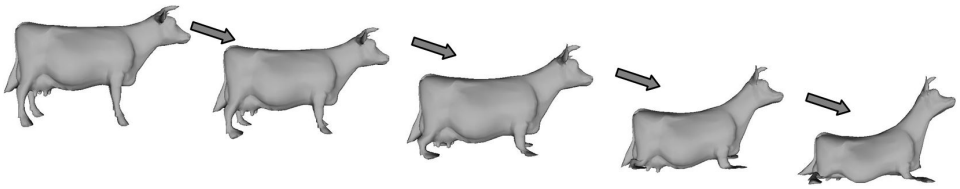
Figure 3: Visualization of clustering results.



(a) Reconstruction of wavelet coefficients $C_0^T + C_1^T$ of the cow (C1) sequence

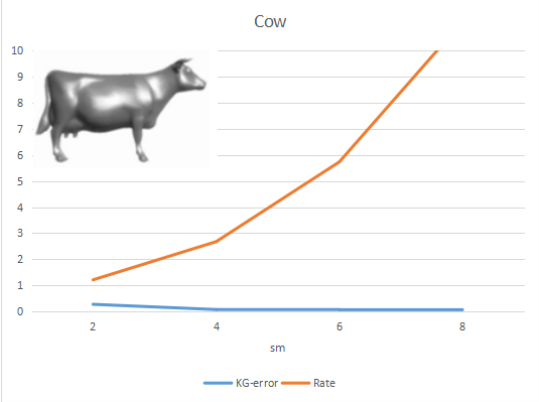


(b) Reconstruction of wavelet coefficients $C_0^T + C_1^T + C_2^T$ of the cow (C1) sequence

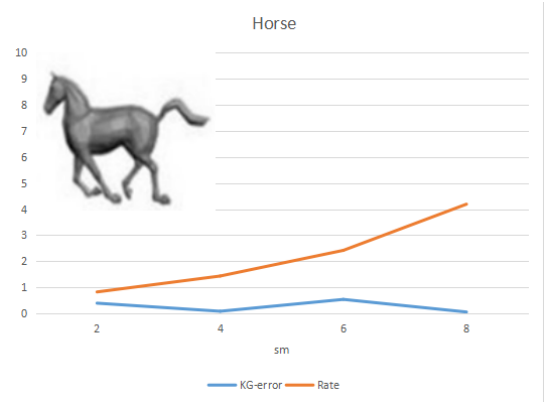


(c) Complete reconstruction of the cow (C1) sequence

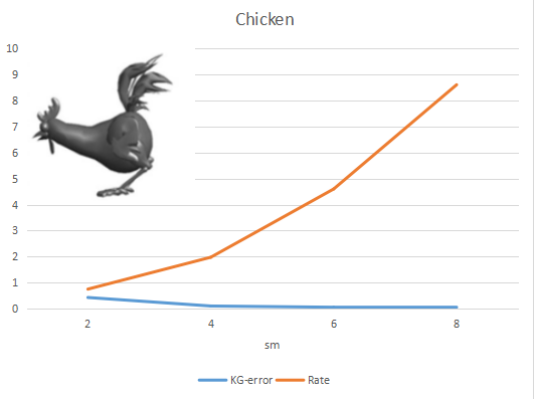
Figure 4: Progressive reconstruction of the cow animation.



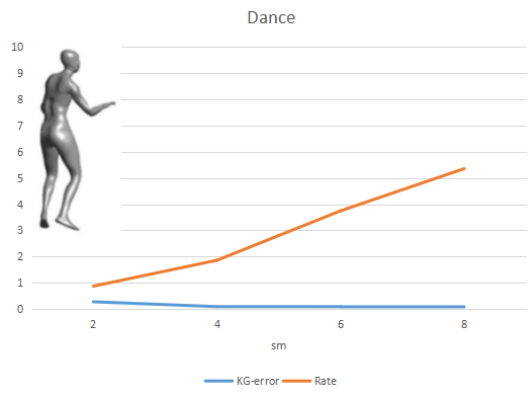
(a) Rate-KG error curve for the cow sequence



(b) Rate-KG error curve for the horse sequence

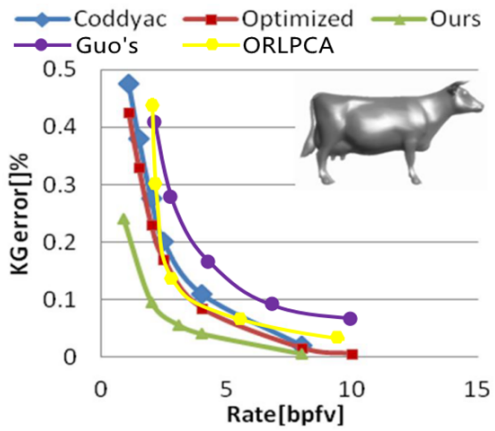


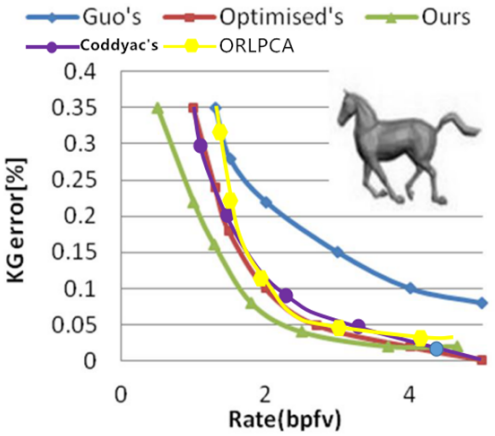
(c) Rate-KG error curve for the chicken sequence



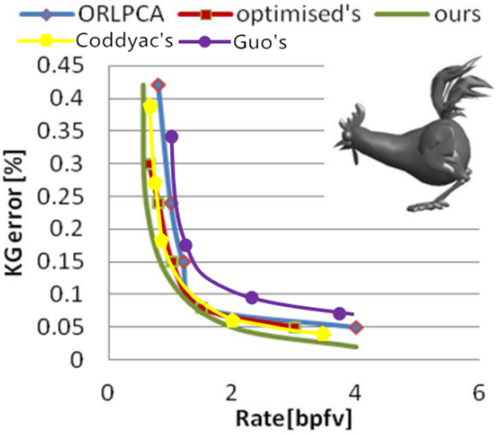
(d) Rate-KG error curve for the dance sequence

Figure 5: Rate-KG error of compression

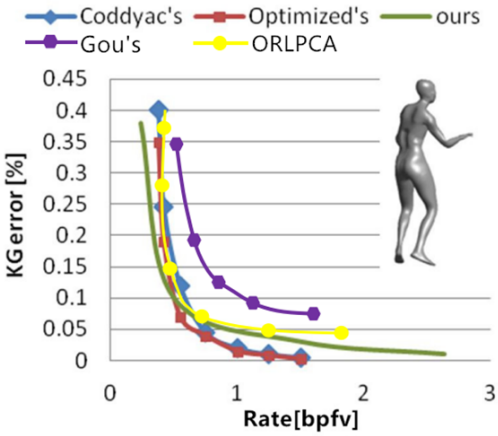




(a) Rate-distortion curve for the horse sequence



(b) Rate-distortion curve for the chicken sequence



(c) Rate-distortion curve for the dance sequence

Figure 6: Rate-distortion of compression with other algorithms